

ISIS-II PL/M-80 V3.1 COMPILATION OF MODULE REN
 OBJECT MODULE PLACED IN RENAME.OBJ
 COMPILER INVOKED BY: PLM80 RENAME.PLM PAGESWIDTH(100) DEBUG OPTIMIZE

```

1      $ TITLE('CP/M 3.0 --- REN ')
      ren:
      do;

      /*
      Copyright (C) 1982
      Digital Research
      P.O. Box 579
      Pacific Grove, CA 93950
      */

      /*
      Revised:
      19 Jan 80 by Thomas Rolander
      14 Sept 81 by Doug Huskey
      23 June 82 by John Knight
      29 Sept 82 by Thomas J. Mason
      03 Dec 82 by Bruce Skidmore
      */

2 1    declare
      mpmpproduct literally '01h', /* requires mp/m */
      cpaversion literally '30h'; /* requires 3.0 cp/m */

3 1    declare
      true  literally 'OFFh',
      false literally '0',
      forever literally 'while true',
      lit  literally 'literally',
      proc  literally 'procedure',
      dcl  literally 'declare',
      addr  literally 'address',
      cr  literally '13',
      lf  literally '10',
      ctrlc literally '3',
      ctrlx literally '18h',
      bksp  literally '8',
      dcnt$offset  literally '45h',
      searcha$offset  literally '47h',
      searchl$offset  literally '49h',
      hashl$offset  literally '00h',
      hash2$offset  literally '02h',
      hash3$offset  literally '04h';

4 1    declare plm label public;

```

COPYRIGHT ©1982
 DIGITAL RESEARCH
 P.O. BOX 579
 PACIFIC GROVE, CA 93950

SER. # _____

```

/*****
*
* B D O S  INTERFACE
*
*****/

```

```

*
*****/

5 1  mon1:
    procedure (func,info) external;
6 2      declare func byte;
7 2      declare info address;
8 2      end mon1;

9 1  mon2:
    procedure (func,info) byte external;
10 2      declare func byte;
11 2      declare info address;
12 2      end mon2;

13 1 mon3:
    procedure (func,info) address external;
14 2      declare func byte;
15 2      declare info address;
16 2      end mon3;

17 1 declare cmdrv   byte   external; /* command drive */
18 1 declare fcb (1)  byte   external; /* 1st default fcb */
19 1 declare fcb16 (1) byte   external; /* 2nd default fcb */
20 1 declare pass0    address external; /* 1st password ptr */
21 1 declare len0     byte   external; /* 1st passwd length */
22 1 declare pass1    address external; /* 2nd password ptr */
23 1 declare len1     byte   external; /* 2nd passwd length */
24 1 declare tbuff (1) byte   external; /* default dma buffer */

```

```

/*****
*
*      B D O S  Externals
*
*****/

```

```

25 1  read$console:
    procedure byte;
26 2      return mon2 (1,0);
27 2      end read$console;

28 1  conin:
    procedure byte;
29 2      return mon2(6,0ffh);
30 2      end conin;

31 1  printchar:
    procedure (char);
32 2      declare char byte;
33 2      call mon1 (2,char);
34 2      end printchar;

35 1  print$buf:
    procedure (buffer$address);
36 2      declare buffer$address address;

```

COPYRIGHT © 1982
 DIGITAL RESEARCH
 P. O. BOX 579
 PACIFIC GROVE, CA 93950

SER. # _____

```
37 2      call mon1 (9,buffer$address);
38 2      end print$buf;

39 1      read$console$buf:
          procedure (buffer$address,max) byte;
40 2          declare buffer$address address;
41 2          declare new$max based buffer$address byte;
42 2          declare max byte;
43 2          new$max = max;
44 2          call mon1 (10,buffer$address);
45 2          buffer$address = buffer$address + 1;
46 2          return new$max; /* actually number of chars input */
47 2      end read$console$buf;

48 1      check$con$stat:
          procedure byte;
49 2          return mon2 (11,0);
50 2      end check$con$stat;

51 1      version: procedure address;
          /* returns current cp/m version # */
52 2          return mon3(12,0);
53 2      end version;

54 1      search$first:
          procedure (fcb$address) byte;
55 2          declare fcb$address address;
56 2          return mon2 (17,fcb$address);
57 2      end search$first;

58 1      search$next:
          procedure byte;
59 2          return mon2 (18,0);
60 2      end search$next;

61 1      delete$file:
          procedure (fcb$address);
62 2          declare fcb$address address;
63 2          call mon1 (19,fcb$address);
64 2          end delete$file;

65 1      rename$file:
          procedure (fcb$address) address;
66 2          declare fcb$address address;
67 2          return mon3 (23,fcb$address);
68 2          end rename$file;

69 1      setdma: procedure(dma);
70 2          declare dma address;
71 2          call mon1(26,dma);
72 2          end setdma;

          /* Off => return BDOS errors */
73 1      return$errors:
          procedure(mode);
74 2          declare mode byte;
75 2          call mon1 (45,mode);
```

COPYRIGHT ©1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950

SER. # _____

```

76 2      end return$errors;

77 1      declare
          parse$fn structure (
              buff$adr address,
              fcb$adr address);

78 1      parse: procedure (pfcb) address external;
79 2          declare pfcb address;
80 2          end parse;

81 1      declare scbpd structure
          (offset byte,
           set byte,
           value address);

82 1      getscbbyte:
          procedure (offset) byte;
83 2          declare offset byte;
84 2          scbpd.offset = offset;
85 2          scbpd.set = 0;
86 2          return mon2(49,.scbpd);
87 2      end getscbbyte;

88 1      getscbword:
          procedure (offset) address;
89 2          declare offset byte;
90 2          scbpd.offset = offset;
91 2          scbpd.set = 0;
92 2          return mon3(49,.scbpd);
93 2      end getscbword;

94 1      setscbword:
          procedure (offset,value);
95 2          declare offset byte;
96 2          declare value address;
97 2          scbpd.offset = offset;
98 2          scbpd.set = 0FEh;
99 2          scbpd.value = value;
100 2          call mon1(49,.scbpd);
101 2      end setscbword;

```

COPYRIGHT ©1982
 DIGITAL RESEARCH
 P.O. BOX 579
 PACIFIC GROVE, CA 93950

SER. # _____

```

/*****
*
*      GLOBAL VARIABLES
*
*****/

```

/* Note: there are three fcb's used by
this program:

- 1) new\$fcb: the new file name
(this can be a wildcard if it
has the same pattern of question
marks as the old file name)
Any question marks are replaced

with the corresponding filename
character in the old\$fcf before
doing the rename function.

2) cur\$fcf: the file to be renamed
specified in the rename command.
(any question marks must correspond
to question marks in new\$fcf).

3) old\$fcf: a fcf in the directory
matching the cur\$fcf and used in
the bdos rename function. This
cannot contain any question marks.

*/

```
102 1 declare successful lit 'OFFh';
103 1 declare failed      (*) byte data(cr,lf,'ERROR: Not renamed, $'),
      read$only      (*) byte data(cr,lf,'ERROR: Drive read only.$'),
      bad$wildcard (*) byte data('Invalid wildcard.$');
104 1 declare passwd (8) byte;
105 1 declare
      new$fcf$adr address,      /* new name */
      new$fcf based new$fcf$adr (32) byte;
106 1 declare cur$fcf (33) byte; /* current fcf (old name) */
```

```
/******
 *                               *
 *      SUBROUTINES            *
 *                               *
 *****/
```

```
/* upper case character from console */
107 1 crlf: proc;
108 2   call printchar(cr);
109 2   call printchar(lf);
110 2   end crlf;
```

```
/* ----- */
```

```
/* fill string @ s for c bytes with f */
111 1 fill: proc(s,f,c);
112 2   dcl s addr,
      (f,c) byte,
      a based s byte;

113 2   do while (c!=c-1)<255;
114 3     a = f;
115 3     s = s+1;
116 3   end;
117 2   end fill;
```

```
/* ----- */
```

```
/* error message routine */
118 1 error: proc(code);
119 2 declare
```

COPYRIGHT ©1982

DIGITAL RESEARCH

P.O. BOX 579

PACIFIC GROVE, CA 93950

SER. # _____

code byte;

```

120 2    if code = 0 then do;
122 3        call print$buf(,('ERROR: No such file to rename.$'));
123 3        call mon1(0,0);
124 3        end;
125 2    if code=1 then do;
127 3        call print$buf(,(cr,lf,'Disk I/O.$'));
128 3        call mon1(0,0);
129 3        end;
130 2    if code=2 then do;
132 3        call print$buf(,read$only);
133 3        call mon1(0,0);
134 3        end;
135 2    if code = 3 then
136 2        call print$buf(,read$only(15));
137 2    if code = 5 then
138 2        call print$buf(,('Currently Opened.$'));
139 2    if code = 7 then
140 2        call print$buf(,('Bad password.$'));
141 2    if code = 8 then
142 2        call print$buf(,('file already exists$'));
143 2    if code = 9 then do;
145 3        call print$buf(,bad$wildcard);
146 3        call mon1(0,0);
147 3        end;
148 2    end error;

```

COPYRIGHT ©1982
 DIGITAL RESEARCH
 P.O. BOX 579
 PACIFIC GROVE, CA 93950

SER. # _____

/* ----- */

```

/* print file name */
149 1    print$file: procedure(fcbp);
150 2        declare k byte;
151 2        declare typ lit '9';    /* file type */
152 2        declare fnam lit '11'; /* file type */
153 2        declare
            fcbp  addr,
            fcbv  based fcbp (32) byte;

154 2        do k = 1 to fnam;
155 3            if k = typ then
156 3                call printchar(' ');
157 3                call printchar(fcbv(k) and 7fh);
158 3            end;
159 2        end print$file;

```

/* ----- */

```

/* try to rename fcb at old$fcb$adr to name at new$fcb$adr
   return error code if unsuccessful */
160 1    rename:
        procedure(old$fcb$adr) byte;
161 2    declare
        old$fcb$adr address,
        old$fcb based old$fcb$adr (32) byte,
        error$code address,
        code      byte;

```

```

162 2      call move (16,new$fc$adr,old$fc$adr+16);
163 2      call setdma(.passwd);          /* password */
164 2      call return$errors(0FFh);      /* return bdos errors */
165 2      error$code = rename$file (old$fc$adr);
166 2      call return$errors(0);          /* normal error mode */
167 2      if low(error$code) = 0FFh then do;
169 3          code = high(error$code);
170 3          if code < 3 then
171 3              call error(code);
172 3          return code;
173 3          end;
174 2      return successful;
175 2      end rename;

```

```
/* ----- */
```

```

/* upper case character from console */
176 1      ucase: proc(c) byte;
177 2          dcl c byte;

178 2          if c >= 'a' then
179 2              if c < '{' then
180 2                  return(c-20h);
181 2          return c;
182 2          end ucase;

```

```
/* ----- */
```

```

/* get password and place at fcb + 16 */
183 1      getpasswd: proc;
184 2          dcl (i,c) byte;

185 2          call crlf;
186 2          call print$buf(('.Enter password: ','$'));
187 2      retry:
188 2          call fill(.passwd,' ',8);
189 2          do i = 0 to 7;
189 3      nxtchr:
190 3          if (c:=ucase(conin)) >= ' ' then
191 3              passwd(i)=c;
192 3          if c = cr then do;
193 4              call crlf;
194 4              go to exit;
195 4          end;
196 3          if c = ctrlx then
197 3              goto retry;
198 3          if c = bksp then do;
200 4              if i<1 then
201 4                  goto retry;
202 4              else do;
203 5                  passwd(i:=i-1)=' ';
204 5                  goto nxtchr;
205 5              end;
206 4          end;
207 3          if c = ctrlc then
208 3              call mon1(0,0);

```

COPYRIGHT ©1982
 DIGITAL RESEARCH
 P.O. BOX 579
 PACIFIC GROVE, CA 93950

SER. # _____

```

209 3      end;
210 2      exit;
          c = check$con$stat;          /* clear raw I/O mode */
211 2      end getpasswd;

/* ----- */

          /* check for wildcard in rename command */
212 1      wildcard: proc byte;
213 2          dcl (i,wild) byte;

214 2          wild = false;
215 2          do i=1 to 11;
216 3              if cur$fcbl(i) = '?' then
217 3                  if new$fcbl(i) <> '?' then do;
218 4                      call print$buf(,failed);
219 4                      call print$buf(,bad$wildcard);
220 4                      call monl(0,0);
221 4                      end;
222 4                  else
223 3                      wild = true;
224 3                  end;
225 2          return wild;
226 2          end wildcard;

/* ----- */

          /* set up new name for rename function */
227 1      set$new$fcbl: proc(old$fcbl$adr);
228 2          dcl old$fcbl$adr address,
          old$fcbl based old$fcbl$adr (32) byte;
229 2          dcl i byte;

230 2          old$fcbl(0) = cur$fcbl(0); /* set up drive */
231 2          do i=1 to 11;
232 3              if cur$fcbl(i) = '?' then
233 3                  new$fcbl(i) = old$fcbl(i);
234 3              end;
235 2          end set$new$fcbl;

/* ----- */

          /* try deleting files one at a time */
236 1      single$file:
          procedure;
237 2          declare (code,dcnt) byte;
238 2          declare (old$fcbl$adr,savdcnt,savsearcha,savsearchl) addr;
239 2          declare old$fcbl based old$fcbl$adr (32) byte;
240 2          declare (hash1,hash2,hash3) address;

241 2          file$err: procedure(fcbl);
242 3              dcl fcbl address;
243 3              call print$buf(,failed);
244 3              call print$file(fcbl);
245 3              call printchar(' ');
246 3              call error(code);
247 3          end file$err;

```

COPYRIGHT ©1982

DIGITAL RESEARCH

P.O. BOX 579

PACIFIC GROVE, CA 93950

SER. # _____


```

248 2      call setdma(.tbuff);
249 2      if (dcnt:=search$first(.cur$fcf)) = 0fff then
250 2          call error(0);

251 2          do while dcnt <> 0fff;
252 3              old$fcf$adr = shl(dcnt,5) + .tbuff;
253 3              savdcnt = getscbword(dcnt$offset);
254 3              savsearcha = getscbword(searcha$offset);
255 3              savsearchl = getscbword(searchl$offset);
                /* save searched fcf's hash code (5 bytes) */
256 3              hash1 = getscbword(hash1$offset);
257 3              hash2 = getscbword(hash2$offset);
258 3              hash3 = getscbword(hash3$offset); /* saved one extra byte */
259 3              call setnew$fcf(old$fcf$adr);
260 3              if (code:=rename(old$fcf$adr)) = 8 then do;
261 4                  call file$err(new$fcf$adr);
262 4                  call print$buf(.,(' delete (Y/N)?$'));
263 4                  if ucase(read$console) = 'Y' then do;
264 5                      call delete$file(new$fcf$adr);
265 5                      code = rename(old$fcf$adr);
266 5                      end;
                else
267 4                  go to next;
268 4                  end;

269 3              if code = 7 then do;
270 4                  call file$err(old$fcf$adr);
271 4                  call getpasswd;
272 4                  code = rename(old$fcf$adr);
273 4                  end;
274 3              if code <> successful then
275 3                  call file$err(old$fcf$adr);
276 3              else do;
277 4                  call crlf;
278 4                  call print$file(new$fcf$adr);
279 4                  call printchar('=');
280 4                  call print$file(old$fcf$adr);
281 4                  end;
282 3              next:
283 3                  call setdma(.tbuff);
284 3                  call setscbword(dcnt$offset,savdcnt);
285 3                  call setscbword(searcha$offset,savsearcha);
286 3                  call setscbword(searchl$offset,savsearchl);
                /* restore hash code */
287 3                  call setscbword(hash1$offset,hash1);
288 3                  call setscbword(hash2$offset,hash2);
289 3                  call setscbword(hash3$offset,hash3);
290 3                  if .cur$fcf <> savsearcha then /*restore orig fcf if destroyed*/
291 3                      call move(16,.cur$fcf,savsearcha);
292 3                  dcnt = search$next;
293 3                  end;
294 2      end single$file;

                /* ----- */

```

COPYRIGHT ©1982
 DIGITAL RESEARCH
 P.O. BOX 579
 PACIFIC GROVE, CA 93950

SER. # _____

```

297 1      bad$entry: proc;
                /* invalid rename command */

```

```

298 2      call print$buf(,failed);
299 2      call print$buf(,('ERROR: Invalid File.',cr,lf,'$'));
300 2      call mon1(0,0);
301 2      end bad$entry;

```

```
/* ----- */
```

```

302 1  finish$parse: procedure;
303 2      parse$fn.buff$adr = parse$fn.fcb$adr+1; /* skip delimiter */
304 2      parse$fn.fcb$adr = .cur$fcb;
305 2      parse$fn.fcb$adr = parse(,parse$fn);
306 2      call move(8,.cur$fcb+16,.passwd);
307 2      end finish$parse;

```

```
/* ----- */
```

```

308 1  input$found: procedure (buffer$adr) byte;
309 2      declare buffer$adr address;
310 2      declare char based buffer$adr byte;
311 2      do while (char = ' ') or (char = 9); /* tabs & spaces */
312 3          buffer$adr = buffer$adr + 1;
313 3      end;
314 2      if char = 0 then /* eoln */
315 2          return false; /* input not found */
316 2      else
317 2          return true; /* input found */
318 2      end input$found;

```

COPYRIGHT ©1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950

SER. # _____

```
/* ----- */
```

```

/*****
*
*      M A I N   P R O G R A M
*
*****/

```

```

318 1  declare ver address;
319 1  declare i byte;
320 1  declare no$chars byte; /* number characters input */
321 1  declare second$string$ptr address; /* points to second filename input */
322 1  declare ptr based second$string$ptr byte;
323 1  declare last$dseg$byte byte
324 1      initial (0);

324 1  plm:
325 1      ver = version;
326 1      if (low(ver) < cpmversion) or (high(ver) = mpmpproduct) then do;
327 2          call print$buf(,('Requires CP/M 3.0', '$'));
328 2          call mon1(0,0);
329 2      end;

330 1      parse$fn.buff$adr = .tbuff(1);
331 1      new$fcb$adr, parse$fn.fcb$adr = .fcb;
332 1      if input$found(,tbuff(1)) then do;
333 2          if (parse$fn.fcb$adr:=parse(,parse$fn)) <> 0FFFFh then

```

```

335 2      call finish$parse;
336 2      end;
337 1      else do;

          /* prompt for files */
338 2      call print$buf(,('Enter New Name: $'));
339 2      no$chars = read$console$buf(,tbuff(0),40);
340 2      if no$chars <= 0 then do;
342 3          call print$buf(,('ERROR: Incorrect file specification.',cr,lf,$'));
343 3          call mon1(0,0);
344 3          end; /* no$char check */

345 2      tbuff(1)= ' '; /* blank out nc field for file 1 */
346 2      second$string$ptr = ,tbuff(no$chars + 2);
347 2      call crlf;

348 2      call print$buf(,('Enter Old Name: $'));
349 2      no$chars = read$console$buf(second$string$ptr,40);
350 2      call crlf;
351 2      ptr = ' '; /* blank out mx field */
352 2      second$string$ptr = second$string$ptr + 1;
353 2      ptr = '='; /* insert delimiter for parse */
354 2      second$string$ptr = second$string$ptr + no$chars + 1; /* eoln */
355 2      ptr = cr; /* put eoln delimiter in string */
356 2      parse$fn.buff$adr = ,tbuff(1);
357 2      new$fcb$adr, parse$fn.fcb$adr = ,fcb;
358 2      if (parse$fn.fcb$adr := parse(,parse$fn)) <> 0FFFFh then
359 2          call finish$parse;
360 2      end;
361 1      if parse$fn.fcb$adr = 0FFFFh then
362 1          call bad$entry;
363 1      if fcb(0) <> 0 then
364 1          if cur$fcb(0) <> 0 then do;
366 2              if fcb(0) <> cur$fcb(0) then
367 2                  call bad$entry;
368 2              end;
          else
369 1              cur$fcb(0) = new$fcb(0); /* set drive */
370 1      if wildcard then
371 1          call singlefile;
372 1      else if rename(,cur$fcb) <> successful then
373 1          call singlefile;
          call mon1(0,0);
375 1      end ren;

```

MODULE INFORMATION:

```

CODE AREA SIZE      = 0865H   2149D
VARIABLE AREA SIZE  = 0077H   119D
MAXIMUM STACK SIZE  = 000AH   10D
608 LINES READ
0 PROGRAM ERROR(S)

```

COPYRIGHT ©1982

DIGITAL RESEARCH

P.O. BOX 579

PACIFIC GROVE, CA 93950

SER. # _____

END OF PL/M-80 COMPILATION

0000 RENAME
 0000 MCD80A
 0004 BDISK 0080 BUFF 0080 BUFFA 0050 CMDRV 0000 CPU
 007C CR 006D DOLLA 005C FCB 006C FCB16 005C FCBA
 005C IFCB 005C IFCBA 0003 IOBYTE 0053 LENO 0056 LEN1
 0006 MAXB 0006 MEMSIZ 0005 MON1 0005 MON2 0005 MON2A
 0005 MON3 0000 OFFSET 006E PARMA 0051 PASS0 0054 PASS1
 007F R0 007D RR 007D RRECA 007F RRECO 005C SFCB
 0080 TBUF
 0000 REN
 0C3D MEMORY 02BC PLM 0415 READCONSOLE 041E CONIN
 0427 PRINTCHAR 0BC6 CHAR 0437 PRINTBUF 0BC7 BUFFERADDRESS
 0447 READCONSOLEBUF 0BC9 BUFFERADDRESS 0BCB MAX
 0468 CHECKCONSTAT 0471 VERSION 047A SEARCHFIRST
 0BCC FCBADDRESS 048A SEARCHNEXT 0493 DELETEFILE 0BCE FCBADDRESS 04A3 RENAMEFILE
 0BD0 FCBADDRESS 04B3 SETDMA 0BD2 DMA 04C3 RETURNERRORS
 0BD4 MODE 0BD5 PARSEFN 0BD9 SCBPD 04D3 GETSCBWORD 0BDD OFFSET
 04EB GETSCBWORD 0BDE OFFSET 0503 SETSCBWORD 0BDF OFFSET 0BE0 VALUE
 0180 FAILED 0197 READONLY 01B1 BADWILDCARD 0BE2 PASSWD
 0BEA NEWFCBADR 0BEC CURFCB 0529 CRLF 0534 FILL 0C0D S
 0C0F F 0C10 C 055F ERROR 0C11 CODE 05F4 PRINTFILE
 0C12 FCBP 0C14 K 062E RENAME 0C15 OLDFCBADR 0C17 ERRORCODE
 0C19 CODE 068E UCASE 0C1A C 06AC GETPASSWD 0C1B I
 0C1C C 06B5 RETRY 06CE NXTCHR 0743 EXIT 074A WILDCARD
 0C1D I 0C1E WILD 07A4 SETNEWFCB 0C1F OLDFCBADR 0C21 I
 07F1 SINGLEFILE 0C22 CODE 0C23 DCNT 0C24 OLDFCBADR 0C26 SAVDCNT
 0C2B SAVSEARCHA 0C2A SAVSEARCHL 0C2C HASH1 0C2E HASH2 0C30 HASH3
 094E FILEERR 0C32 FCBA 08E8 NEXT 096F BADENTRY
 0984 FINISHPARSE 09AB INPUTFOUND 0C34 BUFFERADR 0C36 VER
 0C38 I 0C39 NOCHARS 0C3A SECONDSTRINGPTR
 0C3C LASTDSEGBYTE
 0000 PARSE
 0B2E DBLNK1 0B22 DEBLNK 0AE0 DELIM 0B1B DELIM1 0B1E DELIM2
 0AD8 PADFLD 0A76 PARS81 0A77 PARS82 09E5 PARSE 0A06 PARSE1
 0A34 PARSE2 0A37 PARSE3 0A49 PARSE4 0A4D PARSE5 0A4F PARSE6
 0A63 PARSE7 0A67 PARSE8 0A88 PARSE9 0AB1 PWFLD 0AC4 PWFLD1
 0AA3 SETFD1 0AA5 SETFD2 0AA9 SETFD3 0ABE SETFLD

COPYRIGHT ©1982

DIGITAL RESEARCH

P.O. BOX 579

PACIFIC GROVE, CA 93950

SER. # _____

0000 RENAME#
0000 MCD80A#
0000 REN#

0415 25	0415 26	041E 27	041E 28	041E 29
0427 30	0427 31	042B 33	0436 34	0437 35
043D 37	0446 38	0447 39	044F 43	0456 44
045F 45	0466 46	0468 47	0468 48	0468 49
0471 50	0471 51	0471 52	047A 53	047A 54
0480 56	048A 57	048A 58	048A 59	0493 60
0493 61	0499 63	04A2 64	04A3 65	04A9 67
04B3 68	04B3 69	04B9 71	04C2 72	04C3 73
04C7 75	04D2 76	04D3 82	04D7 84	04DD 85
04E2 86	04EB 87	04EB 88	04EF 90	04F5 91
04FA 92	0503 93	0503 94	050B 97	0511 98
0516 99	0520 100	0528 101	0529 107	0529 108
052E 109	0533 110	0534 111	0541 113	054D 114
0554 115	055B 116	055E 117	055F 118	0563 120
056B 122	0571 123	0579 124	0579 125	0581 127
0587 128	058F 129	058F 130	0597 132	059D 131
05A5 134	05A5 135	05AD 136	05B3 137	05BB 138
05C1 139	05C9 140	05CF 141	05D7 142	05DD 143
05E5 145	05EB 146	05F3 147	05F3 148	05F4 149
05FA 154	0608 155	0610 156	0615 157	0626 158
062D 159	062E 160	0634 162	064D 163	0653 164
0658 165	0663 166	0668 167	0671 169	0678 170
0680 171	0687 172	068B 173	068B 174	068E 175
068E 176	0692 178	069A 179	06A2 180	06A8 181
06AC 182	06AC 183	06AC 185	06AF 186	06B5 187
06C0 188	06CE 189	06DD 190	06EA 191	06F2 193
06F5 194	06F8 195	06F8 196	0700 197	0703 198
070B 200	0713 201	0716 203	0726 204	0729 205
0729 206	0729 207	0731 208	0739 209	0743 210
0749 211	074A 212	074A 214	074F 215	075B 216
076A 217	077A 219	0780 220	0786 221	078E 222
0791 223	0796 224	07A0 225	07A4 226	07A4 227
07AA 230	07B1 231	07BF 232	07CE 233	07E6 234
07F0 235	07F1 236	094E 241	0954 243	095A 244
0962 245	0967 246	096E 247	07F1 248	07F7 249
0805 250	080A 251	0812 252	0824 253	082C 254
0834 255	083C 256	0844 257	084C 258	0854 259
085C 260	086C 262	0874 263	087A 264	0886 266
088E 267	0899 268	089C 269	089F 270	089F 271
08A7 273	08AF 274	08B2 275	08BD 276	08BD 277
08C5 278	08D0 280	08D3 281	08DB 282	08E0 283
08E8 284	08E8 285	08EE 286	08F7 287	0900 288
0909 289	0912 290	091B 291	0924 292	0931 293
0944 294	094A 295	094D 296	096F 297	096F 298
0975 299	097B 300	0983 301	0984 302	0984 303
098B 304	0991 305	099A 306	09AA 307	09AB 308
09B1 311	09CB 312	09D2 313	09D5 314	09DE 315
09E1 316	09E4 317	02B9 324	02C5 325	02DD 327
02E3 328	02EB 329	02EB 330	02F1 331	02FA 332
0304 334	0317 335	031A 336	031D 338	0323 339
032E 340	0337 342	033D 343	0345 344	0345 345
034A 346	0356 347	0359 348	035F 349	036C 350
036F 351	0374 352	037B 353	0380 354	038F 355
0394 356	039A 357	03A3 358	03B6 359	03B9 360
03B9 361	03C6 362	03C9 363	03D1 364	03D9 366
03E3 367	03E6 368	03E9 369	03F0 370	03F7 371
03FD 372	0408 373	040B 374	0413 375	

0000 PARSE#

COPYRIGHT ©1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950

SER. # _____

COPYRIGHT ©1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950

SER. # _____